Multifractal Description of Resource Exhaustion Data in Real Time Operating System

by

Mark Shereshevsky, Jonathan Crowell and Bojan Cukic

July 2001
(Technical Report presented as deliverable for the project "Fractal Analysis of Resource Exhaustion in Real Time Operating System")

Contents

1. Introduction: Statement of the Problem and State of the Art

Software systems are known to suffer from outages due to transient errors. More recently, the phenomenon of "software aging" has been reported which implies that the state of the software system degrades with time (see e.g. [2,3]). This degradation typically manifests in either performance decline (excessive paging and swapping activity etc.) or crash/hang failures or both. It has been observed [2] that such a degradation results, in particular, from the exhaustion of the operating system resources. Some common examples of "software aging" are memory bloating and leaking, fragmentation, data corruption, unreleased file locks etc. Aging has been observed not only in mass scale software, but also in specialized software used in safety critical applications. In order to prevent crashes resulting from the degradation, one needs to occasionally stop the running software, clean its internal sate and then restart. *In order to optimize the timing of such preventive procedure, one needs to be able to forecast the time when the resource exhaustion reaches the critical level.*

Thus, analysis and modeling of the time series which represents the degradation of operating systems resources (e.g. *usedSwapSpace* and *realMemoryFree*) becomes an important task. The time series in question manifest extremely non-linear behavior. The study in [9] describes the behavior of system recourses as a piece-wise linear function of time, where the slope (trend) of the linear function depends on the workload state of the system, while the workload dynamics is modeled as semi-Markov process.

However, in many workload states the resources observed in [9] demonstrate high variance resulting in very broad confidence intervals. The high irregularity and variability of the data in question makes any linear trend model ineffective for the resource exhaustion forecasting, and calls for the implementation of techniques well suited for modeling highly irregular time series.

Using the fractal approach for studying complex systems has proved to be fruitful in mathematics (analysis of singular measures), and in physics (description of highly non-linear phenomena). Recently a strong progress has been made in signal processing, using, for instance, the *multifractal analysis*. The multifractal analysis provides mathematically sophisticated technique for analyzing highly irregular time series. For instance, in the study of network traffic the use of Markov and Poisson models turned out to be inadequate, while multifractal model proved quite effective (see e.g. [7]).

A bibliographic search on the topics related to the proposed research has been carried out to explore the state of the art in this area, as well as to ensure the uniqueness of our research ideas. Earlier work in measurement-based dependability evaluation (see e.g. [4,5]) was mainly based on measurements made at failure times or error times (without continuously monitoring the system's parameters). The first serious attempt to develop a general methodology for estimating trends and times to exhaustion of operating system resources has been made in [2]. A further progress in this direction has been made in [9], where the semi-Markov model is constructed based on workload and resource usage data collected from the UNIX OS. However, the authors of [9] are looking for linear trends in the resource exhaustion data, without accounting for large deviations from the trend, while the large deviations are typical for this kind of data (see the time plots in [9] and in Section 5 of this Report).

2. Data collection

Two main sets of data have been collected from machines at WVU, and a set of data is available that was collected at Duke University.

The first set of data collected from WVU was collected once per second for 488,051 seconds, or for approximately 5 days and 18 hours. The first data point was collected on June 13 at 6:54:33 PM, and the last data point was collected on June 19 at 10:30:25 AM.

The data in this set was collected on machine Naur, a CSEE department server at WVU. Naur has 256 MB of RAM, has dual 333 MHz processors, runs SunOS 5.5.1, and is one of the most heavily used servers in the department. The UNIX utility used to collect the data was "sar –r", which is "system activity reporter" with the option that reports unused memory pages and disk blocks. The only data collected each second was the number of unused memory pages (*realMemoryfree*) and the number of disk blocks available for page swapping (*freeSwapSpace* (or *totalSwapSpace* - *usedSwapSpace*)), and the time. No complimentary data was collected regarding the system load, but trends in the data can be matched against the time of day.

The second set of data was collected 10 times per second from Paladin, a CSEE department print server with 64 MB of RAM and running Windows NT. 1,000,000 data points were collected from June 26, 2:50:50 PM until June 27, 6:47:45 PM. The data collected was the number of free bytes of real memory (*realMemoryFree*), the number of free bytes in the paging file (*freePageBytes*) and the number of free bytes of virtual memory (*freeVirtualMemory*). The collection was done via a fairly simple C++ application.

The data available from Duke University was collected once every 10 minutes for over 3 months from four machines, Rossby, Vergina, Petri, and Jefferson. Data is available on a couple dozen aspects of the machine's state, including the *realMemoryFree* and *usedSwapSpace* as well as several indicators of the system load. We have not begun to analyze this data since we suspect that the 10-minute intervals between collection times will render it of little use for our purposes. At this point in the project we have tested our algorithms mainly on the *realMemoryFree* time series from the set of data collected from Naur.

3. Hölder Exponent as a Characteristic of Fractal Functions

The study of fractal functions (i.e. those whose plots exhibit fractal properties) has proved important in several scientific domains. The examples of processes described by such functions include the heartbeat oscillations, the speech signals, the network traffic, the stock index charts etc. One of the main mathematical tools of studying the fractal functions is the *Hölder exponent*.

The Hölder exponent of a function $f(t)$ at the point t is defined (see e.g. [10] by using the formula:

$$H_f(t) = \liminf_{\delta \to 0} \frac{\log | f(t+\delta) - f(t) |}{\log | \delta |}, \qquad (1)$$

where "lim inf" stands for the lower limit. Clearly the Hölder exponent is always non-negative ($H_f(t) \geq 0$), and it is easily checked that for any "smooth", i.e. differentiable, function $H_f(t) = 1$ at any point t (where the function's derivative is non-zero). On the other hand, extremely non-linear non-differentiable functions with local singularities exhibiting wild oscillations at every scale have Hölder exponents < 1, the stronger the singularity – the smaller (closer to 0) the value of the Hölder exponent.

For example, consider the function $f(t) = \sqrt{|t|} \sin \frac{1}{t}$ pictured on Fig. 1. At the point $t = 0$, around which we see an extreme oscillatory behavior at every scale, it has Hölder exponent $H_f(0) = 0.5$. However, at all other points the function is smooth and has Hölder exponent equal to 1.

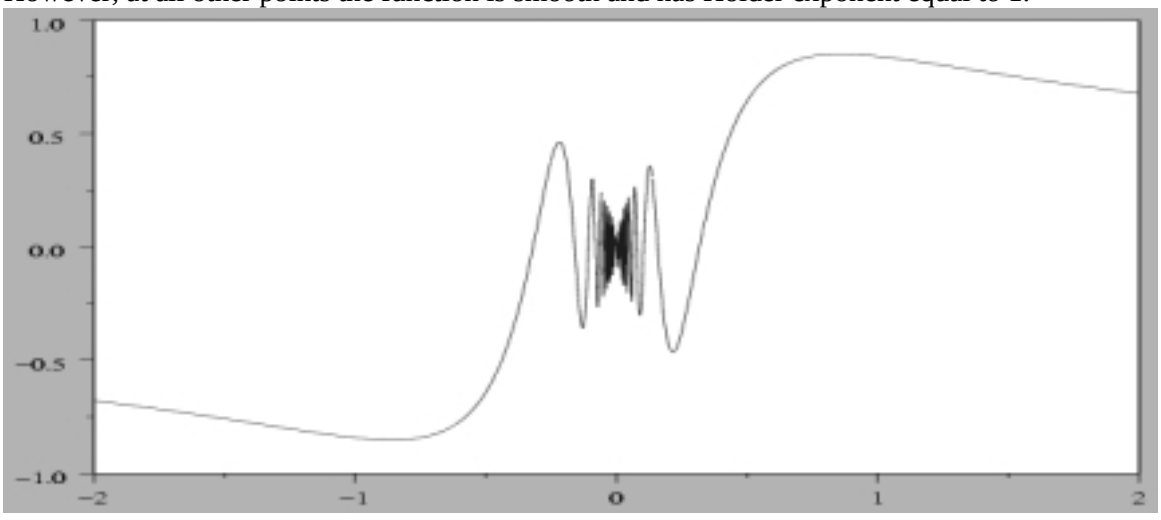

Fig 1. Graph of $f(t) = \sqrt{|t|} \sin \frac{1}{t}$

Fractal functions are characterized by the fact that local singularities are observed not just at individual points but (almost) everywhere. The classical (and oldest) example of a fractal function is the so-called Weirstrass function (see e.g. [12]) given by the formula:

$$w(x) = \sum_{k=0}^{\infty} a^k \cos(2\pi b^k x),$$

where a, b are parameters such that $0 < a < 1 < b$ and $ab \geq 1$. Fig 2 shows the graph of the Weirstrass function with the parameter values $a = 0.5$, $b = 3$.

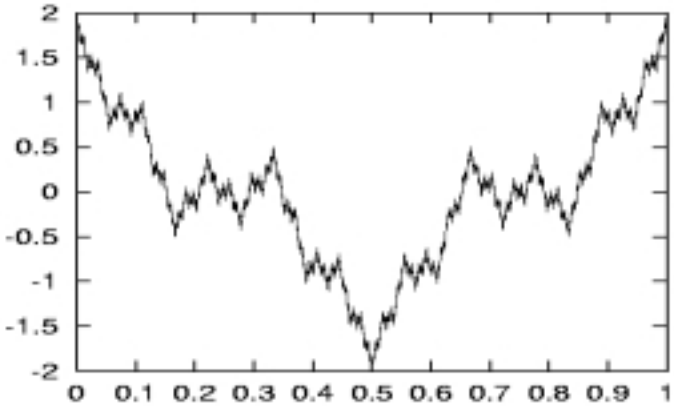

Fig 2. Graph of Weirstrass function

The function is everywhere continuous but nowhere differentiable, i.e. at every point there is a local singularity. Directly from the definition one can compute the Hölder exponent of the Weirstrass function: $H_w(t) = \frac{|\log a|}{\log b}$ for every t, i.e. the local singularity has the same strength everywhere. Such functions are referred to as strictly self-similar or scale invariant. However, generally and especially in experimental setting functions tend to have different singularity strength at different points. This property is called multifractality.

4. Hölder Exponent for a Time Series

The study of experimental time series which have fractal structure is often done by computing and analyzing its Hölder exponent. For example, in [11] this approach was applied to the speech signal analysis and synthesis, and in [13] it was used for diagnosing heart diseases.

In the experimental setting we deal with a discrete sequence of samples (measurements) of a function rather than with the function itself. This means that we cannot apply the definition of the Hölder exponent directly. Based on the formula (1), we have developed the following algorithm for computing the Hölder exponent of the time series.

Suppose $\{y_0, y_1, y_2, \ldots, y_N\}$ is the time series representing a sequence of N measurements of a function $f(t)$, made over a time period $[0, T]$ with a uniform gap between consecutive observations, i.e. y_i represents $f(iT / N)$, $0 \leq i \leq N-1$.

Here is ***an algorithm for estimating Hölder exponent for the time series***.

> *Parameters*: Window width K (a positive integer); weighted regression coefficient α (real $0<\alpha<1$)

For any integer i, $0 \le i \le N-1$:

(*Step 1*) for each integer $k \neq 0$ from $(-K)$ to K, such that $0 \le i+k \le N-1$, compute

$$R_{i,k} := \frac{\log|y_{i+k} - y_i|}{\log(|k|/N)};$$

then

(*Step 2*) for each integer s from 1 to K we compute

$$h_{i,s} := \min\{R_{i,k} : |k| \le s\}$$

(this can be done recursively using the fact that $h_{i,s} = \min\{R_{i,s}, R_{i,-s}, h_{i,s-1}\},\ s = 2,...,K$);

finally,

(*Step 3*) to estimate the limit in the definition of Hölder exponent, perform the weighted regression of the approximations $h_{i,s}$:

$$H_i := \frac{1-\alpha}{1-\alpha^K}(h_{i,1} + \alpha h_{i,2} + ... + \alpha^{K-1} h_{i,K}).$$

The value H_i is an estimate of the Hölder exponent at the ith data point, i.e. $H_f(iT/N)$.

We have written a code in Java implementing the above algorithm. To test the performance of the algorithm we decided to run it on a signal whose Hölder exponent is exactly known. For this purpose we considered the class of so-called *generalized Weirstrass function* (GWF) which have the predetermined Hölder exponent at every point. Given a continuous function $s(t)$ satisfying $0 \le s(t) \le 1$, the corresponding GWF is given by the formula

$$GWF_s(t) = \sum_{j=1}^{\infty} 3^{-j \cdot s(t)} \cdot \sin 3^j t \quad . \qquad (2)$$

Hölder exponent of a GWF at the point t is known to be exactly $s(t)$ (see e.g. [10]). Thus we can chose any function $0 \le s(t) \le 1$ we like, generate a time series by computing the corresponding GWF at regularly spaced values of t. Then we apply our algorithm to extract the Hölder exponents and compare the results with the known answer, namely $s(t)$.

For our testing we first took the generalized Weirstrass function generated by $s(t) = |\sin(5\pi t)|$. The graphs in Fig 3 show both the function $s(t)$ and the corresponding $GWF_s(t)$. We then ran our algorithm for calculating the Hölder exponents with the parameters K=100 and α = 0.9. For comparison, we also computed the Hölder exponents using the *alphagifs* routine from the FracLab library (see [14]) developed by H. Daoudi at INRIA, France. We were pleasantly surprised to find that our algorithm produces much more precise estimates of the Hölder exponent. The results of Hölder exponent estimation using our algorithm and FracLab algorithm are shown in Fig 4. We also carried out the same testing procedure on the GWF generated by other, more complex functions, and again were quite satisfied with the performance of our algorithm.

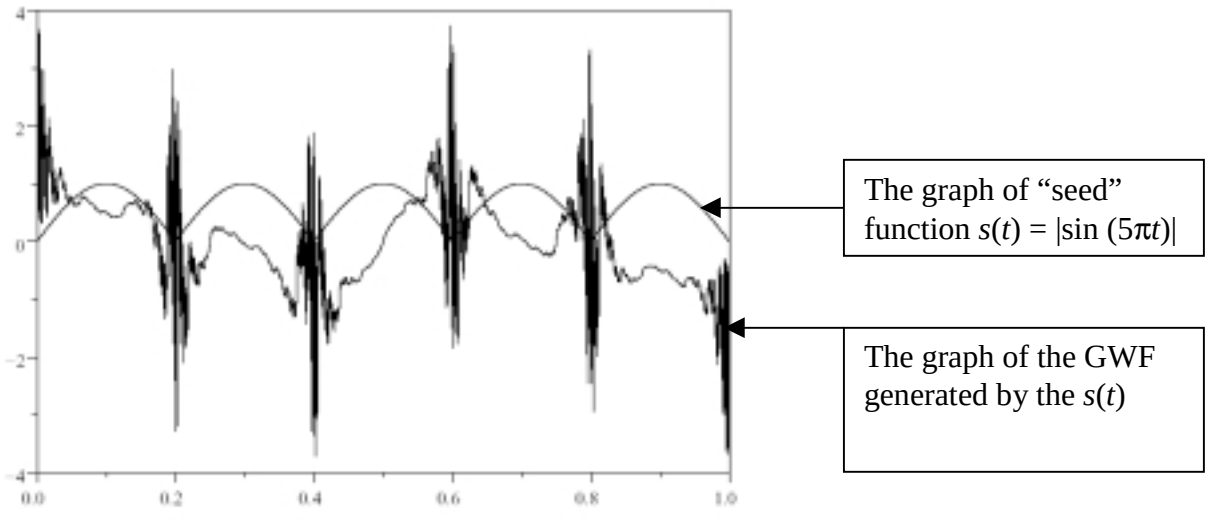

Fig 3. Plots of $s(t) = |\sin(5\pi t)|$ and the corresponding $GWF_s(t)$

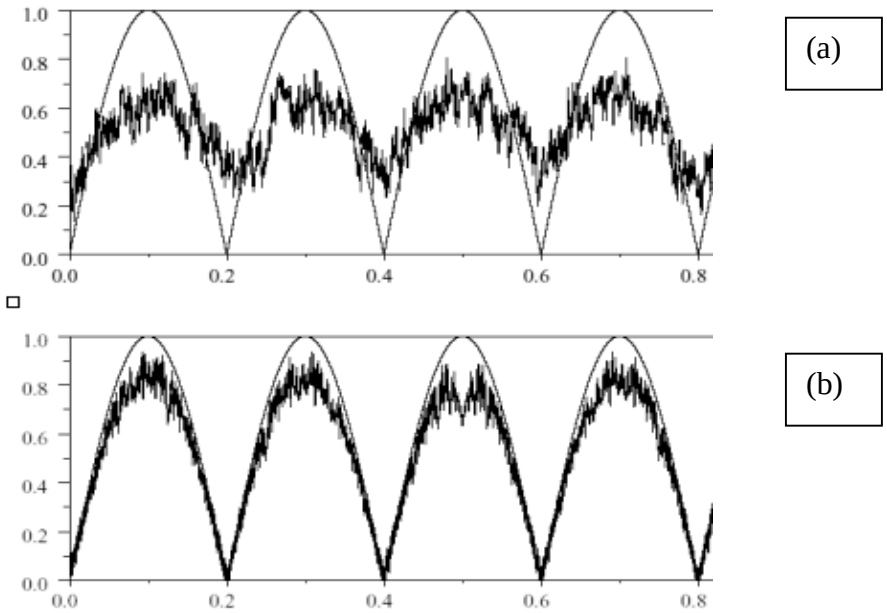

Fig 4. Hölder exponent estimates for $GWF_s(t)$ from Fig 3 obtained (a) using *FracLab*; (b) using our algorithm, versus the actual values

5. Computing and Analyzing Hölder Exponent for Memory Resources Data

Since the testing results described in the previous section gave us confidence in the performance of our algorithm for estimating the Hölder exponent of a time series, we have proceeded to implement it to calculate the Hölder exponents for the operating system memory resources data we collected.

Below we present the results of two experiments performed on the *realMemoryFree* time series from the *Naur* machine. The first one (Fig 5a) is based on a three-hour set of data collected early morning hours: from 2:30 AM to 5:30 AM. The second one (Fig 5b) is based on another three-hour set of data, but collected in the midday: from 12 noon to 3 PM. The measurements have been taken once per second resulting in 10,800 points in each time series. The graphs presented in Fig 5a contain the *realMemoryFree* data plotted on top and their Hölder exponent plotted below. The *realMemoryFree* data have been rescaled (normalized) so that the value 1 on Fig 5a corresponds to the value *realMemoryFree* = 2043, and the value 1 on Fig 5b corresponds to the value *realMemoryFree* = 5033.

Both time series demonstrate evidently multifractal behavior. Since the Hölder exponent plots are extremely chaotic, we have also computed the histograms of the Hölder exponents for both time series, in order to visualize their distribution over the range [0, 1] where they take their values (Fig 6a and 6b).

The histograms show clear differences in the shape of Hölder exponents distribution for these two time series. The midday hours (Fig 6b) which, of course, are characterized by more intensive usage of the system (higher workload), yield *realMemoryFree* data with very few Hölder exponents in the upper part of the range (> 0.8), i.e. very few points where data are relatively smooth. On the other hand, the Hölder exponent histogram of the *realMemoryFree* time series obtained in the night hours (Fig 6a) is quite "fat" in the upper part of the range (> 0.8), i.e. significant portions of the signal are relatively "smooth". Note that the extremely low values of Hölder exponent (very strong singularities) are rare in both graphs. What appears very interesting is that the time series corresponding to the *higher workload* produces a less dispersed spectrum of Hölder exponents which are mainly concentrated in the [0.4, 0.6] range. This is sign of *higher self-similarity* (a perfectly self-similar signal has a strictly constant Hölder exponent).

6. Conclusion and Future Work

The theoretical and experimental results can be summarized as follows:

- we have studied the Hölder exponent as an effective tool for multifractal analysis of the data, and developed an algorithm for estimating the Hölder exponent of an experimental data;
- we have tested the algorithm and found its performance very robust;
- we have collected the memory resource data from a WVU machine, and also obtained the data from the Duke University research group;
- we have run our algorithm for estimating the Hölder exponent on several sets of the *realMemoryFree* resource data from the *Naur* machine;
- we have analyzed the obtained Hölder exponent plots and found the shape of the histogram of the Hölder exponent differs markedly for data corresponding to different workload regimes of the operating system.

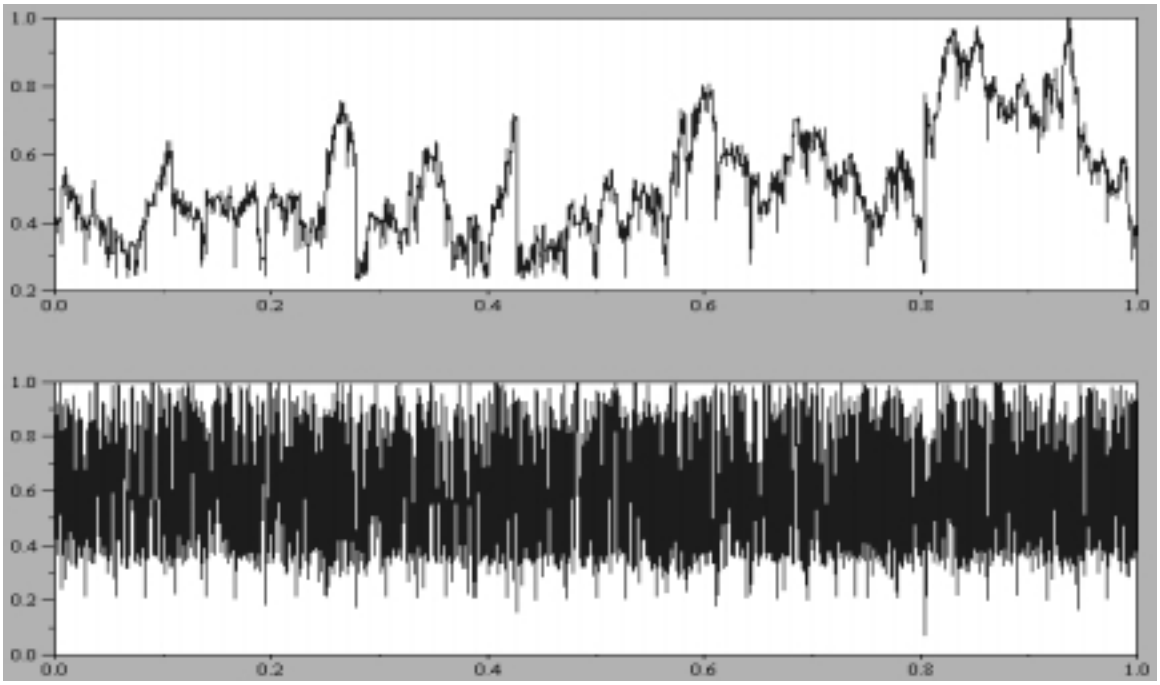

Fig 5a. Upper plot: the *realMemoryFree* data from *Naur* taken once per second for 3 hrs: from 2:30 AM to 5:30 AM (low workload); Lower plot: the Hölder exponents for the data.

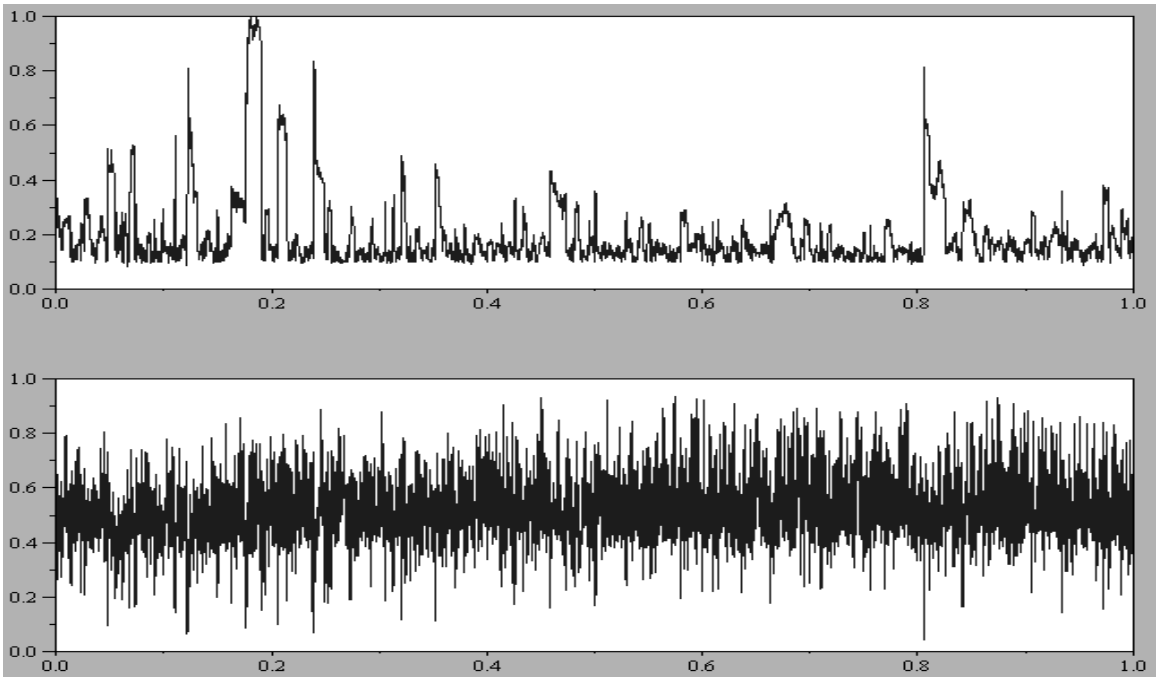

Fig 5b. Upper plot: the *realMemoryFree* data from *Naur* taken once per second for 3 hrs: from 12 noon to 3 PM (high workload); Lower plot: the Hölder exponents for the data.

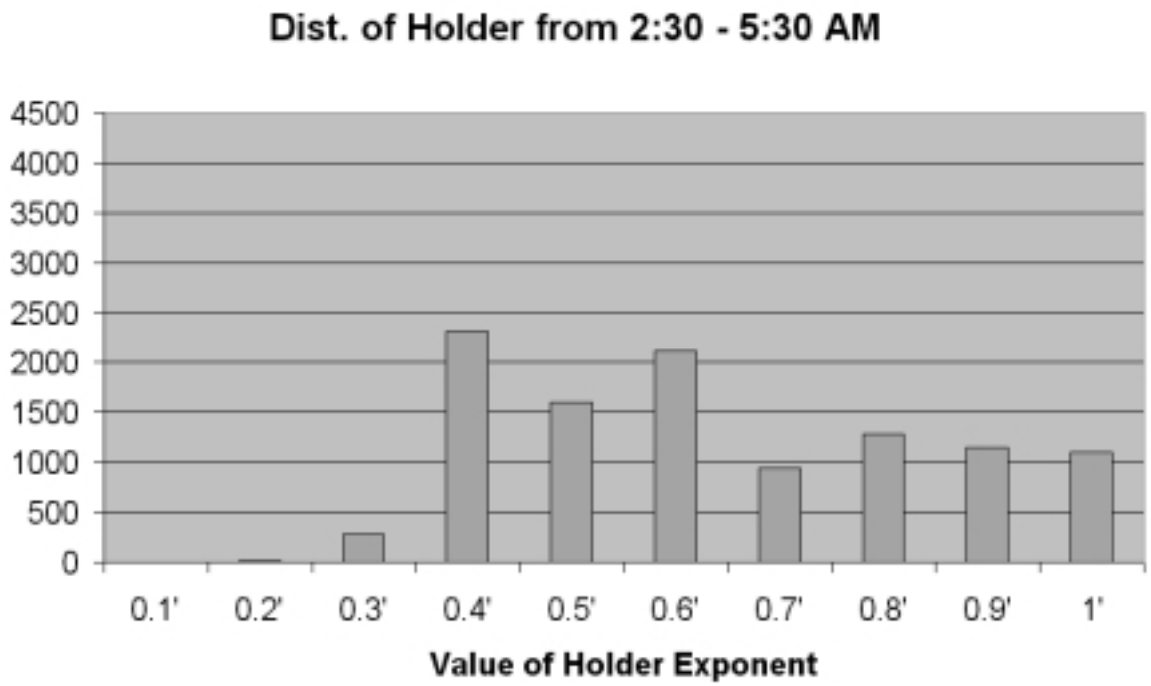

Fig 6a. The histogram (distribution) of the values of Hölder exponent for *realMemoryFree* data from *Naur* taken once per second for 3 hrs: from 2:30 AM to 5:30 AM (low workload).

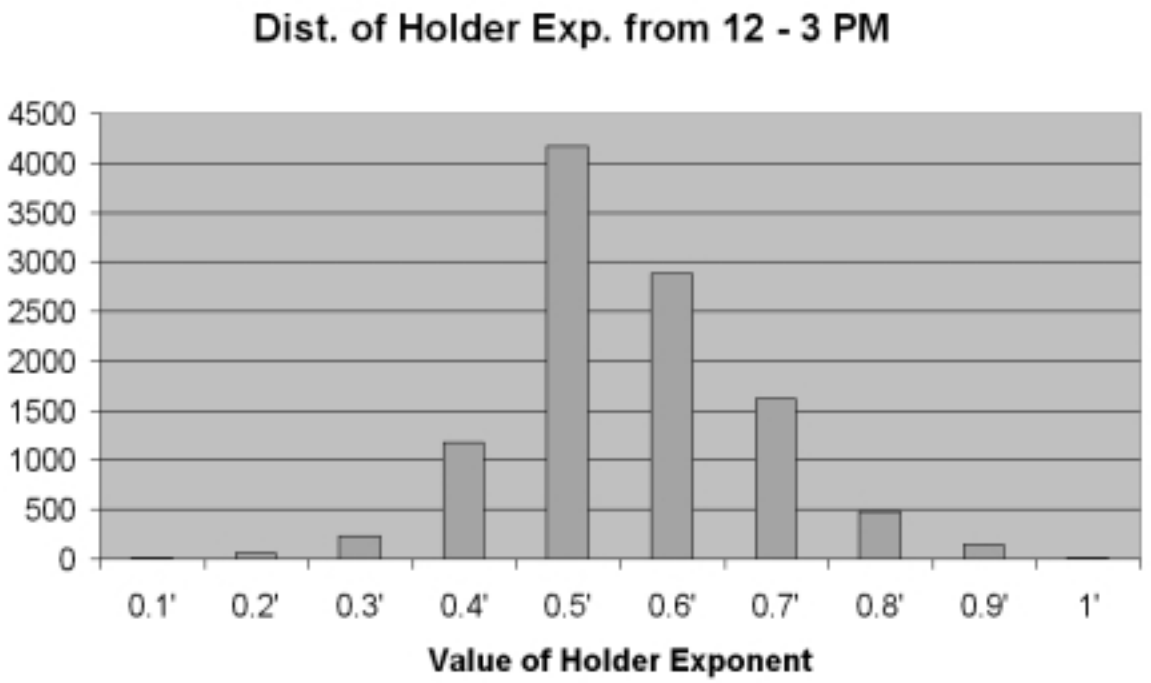

Fig 6b. The histogram (distribution) of the values of Hölder exponent for *realMemoryFree* data from *Naur* taken once per second for 3 hrs: from 12 noon to 3 PM (high workload).

In our further research we plan to address (among others) the following issues:

- modeling of our data using the GIFS (generalized iterated functional systems);
- wavelet analysis of our data;
- fractal dimension of the data;
- correlation between fractal characteristics of the memory resource data and malfunction /failure events

References

1. M. Crovella and A. Bestavros: "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes". *IEEE/ACM Trans. on Nettworking*, 5(6), (1997)

2. S. Garg, A. von Moorsel, K. Vaidyanathan and K. Trivedi: "A Methodology for Detection and Estimation of Software Aging". In *Proc. of 9th Intnl. Symp. On Software Reliability Engineering*, pp. 282-292, Paderborn, Germany (1998)

3. Y.Huang, C.Kintala, N.Kolettis and N.D.Fulton: "Software Rejuvenation: Ananlysis, Module and Applications". In *Proc. of 25th Symp. On Fault Tolerant Computer Systems*, pp. 381-390, Pasadena, CA (1995)

4. R.K. Iyer, D.J.Rossetti, "Effect of System Workload on Operating System Reliability: A Study on IBM 3081". *IEEE Trans. on Software Engineering*, SE-11(12), pp.1438-1448 (1985).

5. R.A. Maxion and F.E. Feather: "A Case Study of Ethernet Anomalies in a Distributed Computing Environment". *IEEE Trans. on Software Reliability*, 39(4) (1990)

6. E. Peters, Fractal Market Analysis, John Wiley, New York, 1994

7. R. Riedi and J Levy Vehel: "Multifractal Properties of TCP Traffic: a Numerical Study", Technical report, DSP Group, Rice University (1997).

8. D. Schertzer, S. Lovejoy: Scaling, Fractals and Non-Linear Variability in Geophysics, Kluwer, Dordrecht-Boston, 1991 (318 pp)

9. K. Vaidyanathan and K. Trivedi: "A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems", In *10th International Symposium on Software Reliability Engineering, Boca Raton, FL* (1999)

10. J. Levy Vehel, and K. Daoudi: "Generalized IFS for Signal processing". *IEEE DSP Workshop*, Loen, Norway, Sept. 1-4, 1996

11. J. Levy Vehel, and K. Daoudi: "Speach Signal Fractal Modeling Based on Local Regularity Analysis". *IASTED/IEEE International Conference on Signal and Image Processing* (SIP'95). Las Vegas, Nov. 20-23, 1995

12. G.H. Hardy, ``Weierstrass's Non-Differentiable Function", *Trans. Amer. Math. Soc.* 17, 301-325, 1916

13. Z. Struzik: "Multifractality in Human Heartbeat Dynamics", *Nature*, Vol. 399, pp. 461-465, June 1999

14. http://www-rocq.inria.fr/fractales/